Appendix for
iDAG: Invariant DAG Searching for Domain Generalization

A. Preliminary Setting

A.1. Notations and Terminology

Table A.1. List of notations.

	Index
i, j	Variable element (matrix element) index
	Variable
$\boldsymbol{x}$	Observational data input
y	Label of the data
$\boldsymbol{z}$	Bottleneck features
$\boldsymbol{v}$	Abstract factors corresponding to nodes in DAG
ϵ_i	Mutually independent exogenous noise variable in generating v_i
$\mathbf{Pa}_i$	Set of causal parents of i-th factor
$\boldsymbol{A}$	Adjacent matrix for DAG representation
$\boldsymbol{P}^{\text{tol}}$	Pairwise total effect matrix, $P^{\text{tol}}_{i,j}$ indicates the total effect $i \to j$
$\boldsymbol{\nu}^e_c$	In-domain prototype variable corresponding to domain e, class c
$\boldsymbol{\mu}_c$	Cross-domain prototype variable corresponding to class c
$\mathcal{B}_\nu$	A set of in-domain prototypes which is updated by original bottleneck features $\boldsymbol{z}^e$
$\mathcal{B}_\mu$	A set of cross-domain prototypes which is updated by invariant (masked) features $\boldsymbol{z}^e_y$
	Function and Hyper-parameter
g	Potential invertible function which control the v_i generating
ϕ	Featurizer which maps data to features
ω	Classifier which maps features to label
λ	Lagrangian multiplier
c	Acyclic penalty parameter
α	Threshold parameter for penalty c updating
η	Acyclic penalty scale factor
λ_{l1}	Sparsity weights for adjacent matrix $\boldsymbol{A}$

A.2. Dataset Details

CMNIST [3] contains images adapted from the MNIST and the binary labels are given by whether the image digit is larger than 5. The digits are colored either red or green so that each color strongly correlates with the label. Additionally, the true labels are flipped with probability 25% as label noise to enhance the spurious correlation between colors and labels. This dataset has 2 classes for each domain, which amounts to 50,000 images of size (2, 14, 14).

PACS [23] is a conventional domain generalization benchmark for image classification. It contains 9,991 images with 7 classes including {dog, elephant, giraffe, guitar, horse, house, person} from 4 domains including {art, cartoons, photos, sketches}.

OfficeHome [53] is a conventional domain generalization benchmark. It contains four domains: {Art, Clipart, Product, Real World}. This dataset has 65 classes for each domain, which amounts to 15,588 images.

DomainNet [38] is a large conventional domain generalization benchmark. It contains six domains {clipart, infograph, painting, quickdraw, real, sketch}. This dataset has 345 classes for each domain, which amounts to 586,575 images of size (3, 224, 224).

B. Motivating Example

B.1. Synthetic Example

In this section, we extend the synthetic experiments and provide further in-depth analysis. Following the data-generating process definition introduced by [3, Arjovsky et al.] we simplify the data-generating process as following scalar formulation:

$$z_y^e \leftarrow \epsilon_x^e, \qquad \epsilon_x^e \sim \mathcal{N}(0, (\sigma_x^e)^2), \tag{17}$$

$$y^e \leftarrow \xi_y z_y^e + \epsilon_y, \qquad \epsilon_y \sim \mathcal{N}(0, \sigma_y^2), \tag{18}$$

$$z_s^e \leftarrow \xi_s^e y^e + \epsilon_s^e, \qquad \epsilon_s^e \sim \mathcal{N}(0, (\sigma_s^e)^2), \tag{19}$$

where ξ_y is underlying weight parameter for invariant features, ξ_s^e is underlying weight parameter for spurious features, $\sigma_x^e, \sigma_y, \sigma_s^e$ are variances of domain-specific additive Gaussian noises. In the sequel, we first discuss why the ordinary ERM cannot produce unbiased estimates in the presence of spurious factors. Then, we show how our DAG modeling is able to recognize spurious relations and leads to unbiased estimations.

Emperical risk minimization perspective. For notation simplicity, denote $\boldsymbol{z}^e = [z_y^e, z_s^e]^\top$ and $\tilde{\xi}_y, \tilde{\xi}_s^e, \tilde{\boldsymbol{\xi}}$ represent the estimated parameters from observation data. Here we illustrate three cases of potential estimation results:

- **Case 1**: Regress from z_y^e to obtain $\hat{\boldsymbol{w}}^\top = [\tilde{\xi}_y, 0]$;
- **Case 2**: Regress from z_s^e to obtain $\hat{\boldsymbol{w}}^\top = [0, \tilde{\xi}_s^e]$;
- **Case 3**: Regress from z_y^e, z_s^e to obtain $\hat{\boldsymbol{w}}^\top = \tilde{\boldsymbol{\xi}}$.

In the general case, we are not given the prior identification information of which portion of observational data $\boldsymbol{z}^e$ is true causal latent. Following the ERM protocol, by using Least Squares Regression, the estimation of $\hat{\boldsymbol{w}}$ is composed of two portions $\hat{w}_y$ and $\hat{w}_s$, that is

$$\begin{aligned} \hat{y}^e &= \begin{bmatrix} \hat{w}_y & \hat{w}_s \end{bmatrix} \cdot \begin{bmatrix} z_y^e \\ z_s^e \end{bmatrix} + \epsilon \\ &= \begin{bmatrix} \hat{w}_y & \hat{w}_s \end{bmatrix} \cdot \begin{bmatrix} \epsilon_x \\ \xi_s^e(\xi_y \epsilon_x + \epsilon_y) + \epsilon_s^e, \end{bmatrix} + \epsilon, \end{aligned} \tag{20}$$

let $\boldsymbol{Z} = [\boldsymbol{z}_1^{e\top}, \boldsymbol{z}_2^{e\top}, \dots]^\top$ then we have,

$$\mathcal{R}(\boldsymbol{w}) = \frac{1}{2}(\boldsymbol{Z}\boldsymbol{w} - \boldsymbol{Y})^\top (\boldsymbol{Z}\boldsymbol{w} - \boldsymbol{Y}), \tag{21}$$

as a solution of Least Squares Regression estimation, $\hat{\boldsymbol{w}}$ is

$$\begin{aligned} \mathbb{E}[\hat{\boldsymbol{w}}] &= (\boldsymbol{Z}^\top \boldsymbol{Z})^{-1} \boldsymbol{Z}^\top \boldsymbol{Y} \\ &= (\boldsymbol{Z}^\top \boldsymbol{Z})^{-1} \boldsymbol{Z}^\top (\boldsymbol{Z}\boldsymbol{w} + \boldsymbol{\epsilon}) \\ &= (\boldsymbol{Z}^\top \boldsymbol{Z})^{-1} \boldsymbol{Z}^\top \boldsymbol{Z}\boldsymbol{w} + (\boldsymbol{Z}^\top \boldsymbol{Z})^{-1} \boldsymbol{Z}^\top \boldsymbol{\epsilon} \\ &= \boldsymbol{w} + (\boldsymbol{Z}^\top \boldsymbol{Z})^{-1} \boldsymbol{Z}^\top \boldsymbol{\epsilon}, \end{aligned} \tag{22}$$

while the insight behind Least Squares Regression estimation is that when the assumptions of Gauss-Markov Theorem are satisfied we have $\mathbb{E}[\hat{\boldsymbol{w}}] = \boldsymbol{w}$. However, the existence of spurious relations in training domains leads to biased estimation $\hat{w}_y = \frac{\xi_y \sigma_s^e}{\sigma_y + \sigma_s^e}$, $\hat{w}_s = \frac{\sigma_y}{\xi_s^e(\sigma_y + \sigma_s^e)}$, that means we can never recover the true data-generating process. A general expectancy is that ϵ_s^e exhibits a sufficient level of variation among different domains in the training distribution. Therefore, one can recover the true parameters from the observation. Unfortunately, in practice, we cannot expect the $\sigma_s^e \to +\infty$ for a variety of reasons, such as selection bias, limited observations, etc.

Directed acyclic graph perspective. In contrast to ERM, we demonstrate that learning with a DAG always results in the first of the following three cases:

- **Case 1**: Regress from z_y^e to obtain $\hat{\boldsymbol{w}}^\top = [\tilde{\xi}_y, 0]$;
- **Case 2**: Regress from z_s^e to obtain $\hat{\boldsymbol{w}}^\top = [0, \tilde{\xi}_s^e]$;
- **Case 3**: Regress from z_y^e, z_s^e to obtain $\hat{\boldsymbol{w}}^\top = \tilde{\boldsymbol{\xi}}$;

When taking all observation data including labels into to directed acyclicity graph regression problem:

$$\begin{bmatrix} z_y^e \\ z_s^e \\ y^e \end{bmatrix} = \begin{bmatrix} 0 & w_{1,2} & w_{1,3} \\ w_{2,1} & 0 & w_{2,3} \\ w_{3,1} & w_{3,2} & 0 \end{bmatrix} \begin{bmatrix} z_y^e \\ z_s^e \\ y^e \end{bmatrix} + \begin{bmatrix} \epsilon_x^e \\ \epsilon_s^e \\ \epsilon_y \end{bmatrix}. \tag{23}$$

When considering the relation between z_s^e and y^e, there are two directions: (i)-$z_s^e \to y^e$ with weight $w_{3,2}$; (ii)-$y^e \to z_s^e$ with weight $w_{2,3}$. Since the final graph is DAG, hence, one of $w_{3,2}, w_{2,3}$ must be zeroed out. Suppose:

$$z_s^e = \xi_s^e y^e + \epsilon_s^e, \tag{24}$$

where y^e and ϵ_s^e are independent. It is easy to check that

$$y^e = \tilde{\xi}_s^e z_s^e + \tilde{\epsilon}_s^e, \tag{25}$$

with $\tilde{\xi}_s^e = \frac{\xi_s^e \text{Var}(y^e)}{(\xi_s^e)^2 \text{Var}(y^e)} \neq \frac{1}{\xi_s^e}$, and $\tilde{\epsilon}_s^e = y^e - \tilde{\xi}_s^e z_s^e$. Therefore, according to the additive noise model, only the direction $y^e \to z_s^e$ can be kept. Finally, with the convergence of the DAG learning, the causal graph of the data-generating process can be recovered as follows:

$$\begin{bmatrix} z_y^e \\ z_s^e \\ y^e \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & \xi_s^e \\ \xi_y & 0 & 0 \end{bmatrix} \begin{bmatrix} z_y^e \\ z_s^e \\ y^e \end{bmatrix} + \begin{bmatrix} \epsilon_x^e \\ \epsilon_s^e \\ \epsilon_y \end{bmatrix}. \tag{26}$$

As a result, by introducing the DAG modeling method, we will be able to model the relationship between $\boldsymbol{z}^e$ and y^e, leading to unbiased estimations.

C. Proofs

C.1. Proof of Theorem 1

Restatement of Theorem 1 If $\mathcal{G}$ matches the common structures of all $\mathcal{G}^e$, then it discards the directed edges that start from domain-private factors $\boldsymbol{v}_r^e$ and identifies the association $v_i \leftrightarrow v_j$ into one correct causal direction.

Proof. The proof consists of two major statements, (i)-Causal direction identification; and (ii)-domain-private factors discards.

In general conditions, the set of factors is fixed, and the edges between them vary in different domains. Hence, a graph $\mathcal{G}^e = (\mathcal{V}, \mathcal{E}^e)$ for a particular domain e consists of a set of vertices $\mathcal{V}$ and a set of edges $\mathcal{E}^e \subseteq \mathcal{V} \times \mathcal{V}$. The $\mathcal{G}^e$ can also be equivalently represented by an adjacent matrix $\boldsymbol{A}^e \in \mathbb{R}^{d \times d}$.

Causal Direction Identification. Let $\boldsymbol{V} \in \mathbb{R}^{d \times n}$ be the values matrix of a set of vertices $\mathcal{V}$, and $\boldsymbol{v} \sim \mathcal{N}(0, \boldsymbol{\Sigma})$ (without the loss of generality, we assume the values are preprocessed by a simple bias term with zero means). The linear DAG model is given by:

$$\boldsymbol{V} = \boldsymbol{A}^e \boldsymbol{V} + \boldsymbol{N}, \tag{27}$$

where $\boldsymbol{N} \in \mathbb{R}^{d \times n}$ denotes the exogenous mutually independent noises. And $\boldsymbol{\epsilon}$ is characterized by the covariance matrix $\boldsymbol{\Omega} = \text{diag}(\sigma_1^2, \ldots, \sigma_d^2)$. Assuming $\boldsymbol{I} - \boldsymbol{A}^e$ is invertible, then we can rewrite the Eq. (27) as:

$$\boldsymbol{V} = (\boldsymbol{I} - \boldsymbol{A}^e)^{-1} \boldsymbol{N}. \tag{28}$$

The precision matrix $\Theta = \boldsymbol{\Sigma}^{-1}$ of $\boldsymbol{V}$ as:

$$\Theta = (\boldsymbol{I} - \boldsymbol{A}^e) \boldsymbol{\Omega}^{-1} (\boldsymbol{I} - \boldsymbol{A}^e)^\top \tag{29}$$

The least squares objectives yields:

$$\mathcal{L}_{\text{rec}}(\boldsymbol{A}^e; \boldsymbol{\Sigma}) = \frac{1}{2} \operatorname{Tr}\left((\boldsymbol{I} - \boldsymbol{A}^e)^\top \boldsymbol{\Sigma} (\boldsymbol{I} - \boldsymbol{A}^e)\right). \tag{30}$$

The log-density function of $\boldsymbol{v}$ is then:

$$\begin{aligned}
\log p(\boldsymbol{v}; \boldsymbol{A}^e, \boldsymbol{\Omega}) &= -\frac{1}{2} \log\det \boldsymbol{\Sigma} - \frac{1}{2} \boldsymbol{v}^\top \Theta \boldsymbol{v} - \frac{d}{2} \log 2\pi && (31)\\
&= -\frac{1}{2} \log\det (\boldsymbol{I} - \boldsymbol{A}^e)^{-1} \boldsymbol{N}\boldsymbol{N}^\top (\boldsymbol{I} - \boldsymbol{A}^e)^{-\top} - \frac{1}{2} \boldsymbol{v}^\top \Theta \boldsymbol{v} - \frac{d}{2} \log 2\pi && (32)\\
&= -\frac{1}{2} \log\det \boldsymbol{\Omega} + \log|\det(\boldsymbol{I} - \boldsymbol{A}^e)| - \frac{1}{2} \boldsymbol{v}^\top (\boldsymbol{I} - \boldsymbol{A}^e) \boldsymbol{\Omega}^{-1} (\boldsymbol{I} - \boldsymbol{A}^e) \boldsymbol{v} + c && (33)\\
&= -\frac{1}{2} \sum_{i=1}^{d} \log \sigma_i^2 + \log|\det(\boldsymbol{I} - \boldsymbol{A}^e)| - \frac{1}{2} \sum_{i=1}^{d} \frac{(v_i - \boldsymbol{A}_i^e \boldsymbol{v})^2}{\sigma_i^2} + c, && (34)
\end{aligned}$$

where $\boldsymbol{A}_i^e \in \mathbb{R}^d$ denotes the i-th row vector of $\boldsymbol{A}$. Eq. (32) means that inputs are generated accompanied with mutually independent noises, which directly introduce the Eq. (28) for covariance matrix $\boldsymbol{\Sigma}$ calculation. And this mutually independent noise constraint lead to an extra regularization term $\log|\det(\boldsymbol{I} - \boldsymbol{A}^e)|$.

Given *i.i.d.* samples $\boldsymbol{V} = [\boldsymbol{v}^{(1)}, \ldots, \boldsymbol{v}^{(n)}] \in \mathbb{R}^{d \times n}$ generated from the ground truth distribution, the average log-likelihood of $\boldsymbol{V}$ is given by:

$$\mathcal{L}_{\text{DAG}}(\boldsymbol{A}^e, \boldsymbol{\Omega}; \boldsymbol{V}) = -\frac{1}{2} \sum_{i=1}^{d} \log \sigma_i^2 + \log|\det(\boldsymbol{I} - \boldsymbol{A}^e)| - \frac{1}{2n} \sum_{i=1}^{d} \sum_{k=1}^{n} \frac{\left(v_i^{(k)} - \boldsymbol{A}_i^e \boldsymbol{v}^{(k)}\right)^2}{\sigma_i^2} + c. \tag{35}$$

To profile out the parameter $\boldsymbol{\Omega}$, we have $\frac{\partial \mathcal{L}}{\partial \sigma_i^2} = 0$ thereby deriving the solution:

$$\hat{\sigma}_i^2 = \frac{1}{n} \sum_{k=1}^{n} \left(v_i^{(k)} - \boldsymbol{A}_i^e \boldsymbol{v}^{(k)}\right)^2, \tag{36}$$

then, take this formulation back, we have:

$$\mathcal{L}(\boldsymbol{A}^e, \hat{\boldsymbol{\Omega}}(\boldsymbol{A}^e); \boldsymbol{V}) = -\frac{1}{2} \sum_{i=1}^{d} \log \left(\sum_{k=1}^{n} \left(v_i^{(k)} - \boldsymbol{A}_i^e \boldsymbol{v}^{(k)}\right)^2 \right) + \log|\det(\boldsymbol{I} - \boldsymbol{A}^e)| + c. \tag{37}$$

The goal is therefore to find the weighted adjacent matrix $\boldsymbol{A}^e$ that maximizes the profile likelihood function $\mathcal{L}(\boldsymbol{A}^e, \hat{\boldsymbol{\Omega}}(\boldsymbol{A}^e); \boldsymbol{V})$.

To find the optimal solution of the loss function, we need to set the derivative to zero and solve for $\boldsymbol{A}^e$. However, the mutually dependent noise condition leads to a non-convex objective, so we may need to use numerical methods to find the optimal solution. Notably, due to the strict constraint that $\boldsymbol{A}^e$ is DAG. Therefore, for arbitrary element pair $A_{i,j}$ and $A_{j,i}$ solution of $\hat{\boldsymbol{A}}^e$, only either of them can hold. Let $\psi(\cdot)$ be the topological ordering function. The decision rule is as follows:

$$\begin{cases}
0 & \text{if } \psi(i) = \psi(j) \text{ or } A_{i,j}^e = A_{j,i}^e = 0; \\
\hat{A}_{i,j}^e \to 0 & \text{if } \mathcal{L}(\boldsymbol{A}^e \mid_{A_{j,i}^e = 0}; \boldsymbol{\Sigma}) > \mathcal{L}(\boldsymbol{A}^e \mid_{A_{i,j}^e = 0}; \boldsymbol{\Sigma}); \\
\hat{A}_{j,i}^e \to 0 & \text{if } \mathcal{L}(\boldsymbol{A}^e \mid_{A_{j,i}^e = 0}; \boldsymbol{\Sigma}) < \mathcal{L}(\boldsymbol{A}^e \mid_{A_{i,j}^e = 0}; \boldsymbol{\Sigma}).
\end{cases} \tag{38}$$

Revisit the definition of the spurious relations in Eq. (1), $y^e = \boldsymbol{\xi}_y \boldsymbol{z}_y^e + \epsilon_y$ and $\boldsymbol{z}_s^e = \boldsymbol{\xi}_s^e y^e + \boldsymbol{\epsilon}_s^e$ and the Proposition 1 in [7], arbitrary reverse the correct direction between elements can always lead to lower likelihood. Therefore, optimizing the expectation on the objective with the DAG constraint always leads to the correct direction.

Discard Domain-private Factors. As soon as the causal directions between factors are determined, the determination of random factors will quickly erode to the null edge. Let $\mathcal{G} = \cap_{e=1}^{E} \mathcal{G}^e$ be the common graph, which means extracting a common set of edges $\mathcal{E} = \cap_{e=1}^{E} \mathcal{E}^e$. For a particular factor v_i in one domain, it has no support in other domains. Accordingly, each graph $\mathcal{G}^e$ is optimally learned, there are no edges sourced from this kind of factor in the graph. As a result, these edges will be dropped from the common graph which is the intersection of all domains. At the time of inference, they will not be considered. □

C.2. Proof of Theorem 2

In practical conditions, there are always massive spurious and domain-private features. Similar result has been derived in [4], we provide a proof sketch. For the sake of the proof, we assume that each domain contains the same number $|\mathcal{D}_L^e|$ of samples. Let $\mathbf{\Sigma}^e$ be the covariance matrix of features in e-th domain, and $\mathbf{\Omega}^e$ be the precision matrix. For any vector $\boldsymbol{v}$, we can incorporate the information of the covariance matrix named Mahalanobis norm $\|\boldsymbol{v}\|_{\boldsymbol{M}} := \sqrt{\boldsymbol{v}^\top \boldsymbol{M}\boldsymbol{v}}$. Let $\|\boldsymbol{A}\|_{\boldsymbol{M}} := \sup_{\boldsymbol{v}} \|\boldsymbol{A}\boldsymbol{M}\boldsymbol{v}\|/\|\boldsymbol{v}\|$.

$$\boldsymbol{A}_{\text{inv}} = \arg\min_{\boldsymbol{A}} \mathbb{E}[\mathcal{L}_{\text{rec}}(\boldsymbol{V}; \boldsymbol{A})]. \tag{39}$$

The loss for a DAG adjacent matrix $\boldsymbol{A}$ with a sample $\boldsymbol{v}$ is:

$$\mathcal{L}_{\text{rec}}(\boldsymbol{v}; \boldsymbol{A}) = \sum_{i=1}^{d} L^2 \left(g_i(\boldsymbol{v}), v_i\right). \tag{40}$$

Assumption 1 (Bounded statistical leverage)**.** *For factors $\boldsymbol{V}$, there exists ρ such that, almost surely:*

$$\frac{\|\mathbf{\Sigma}^{-1/2}\boldsymbol{v}\|}{\sqrt{\mathbb{E}\left[\|\mathbf{\Sigma}^{-1/2}\boldsymbol{v}\|^2\right]}} \leq \rho, \tag{41}$$

Assumption 2 (Subgaussian noise)**.** *There exists finite $\sigma_i \geq 0$ such that, almost surely:*

$$\mathbb{E}[\exp(\eta\epsilon_i)|\mathbf{Pa}_i] \leq \exp(\eta^2\sigma_i^2/2) \quad \forall \eta \in \mathbb{R}, i \in [d]. \tag{42}$$

Assumption 3 (Bounded aprroximation error)**.** *There exist finite $b_0 \geq 0$ such that, almost surely:*

$$\frac{\|\mathbf{\Sigma}^{-1/2}\boldsymbol{v}(\boldsymbol{v} - \boldsymbol{A}\boldsymbol{v})\|}{\sqrt{\mathbb{E}\left[\|\mathbf{\Sigma}^{-1/2}\boldsymbol{v}\|^2\right]}} \leq b_0. \tag{43}$$

Lemma 1 (Excess mean squared error, extended from [4] Proposition 21)**.** *For any $\boldsymbol{A}$,*

$$\mathbb{E}\left[\|\boldsymbol{v} - \boldsymbol{A}\boldsymbol{v}\|^2\right] - \mathbb{E}\left[\|\boldsymbol{v} - \boldsymbol{A}_{\text{inv}}\boldsymbol{v}\|^2\right] = \mathbb{E}\left[\|(\boldsymbol{A} - \boldsymbol{A}_{\text{inv}})\boldsymbol{v}\right] = \|\boldsymbol{A} - \boldsymbol{A}_{\text{inv}}\|_{\mathbf{\Sigma}} \tag{44}$$

$$\hat{\mathbb{E}}\left[\|\boldsymbol{v} - \boldsymbol{A}\boldsymbol{v}\|^2\right] - \hat{\mathbb{E}}\left[\|\boldsymbol{v} - \boldsymbol{A}_{\text{inv}}\boldsymbol{v}\|^2\right] = \|\boldsymbol{A} - \hat{\boldsymbol{A}}_{\text{inv}}\|_{\hat{\mathbf{\Sigma}}}. \tag{45}$$

The same arguments also hold for each $\boldsymbol{A}^e$.

Lemma 2 (Effect of eorrors in $\hat{\mathbf{\Sigma}}$ estimation [4] Theorem 11)**.** *With Assumption 1, for any $\delta \leq \min\{1, de^{-2.6}\}$, with probability at least $1 - \delta$,*

$$\|\mathbf{\Sigma}^{-1/2}(\hat{\mathbf{\Sigma}} - \mathbf{\Sigma})\mathbf{\Sigma}^{-1/2}\| \leq \sqrt{\frac{4\rho^2 d(\ln d + \ln(3/\delta))}{n}} + \frac{2\rho^2 d(\ln d + \ln(3/\delta))}{3n}. \tag{46}$$

Further, if $\|\mathbf{\Sigma}^{-1/2}(\hat{\mathbf{\Sigma}} - \mathbf{\Sigma})\mathbf{\Sigma}^{-1/2}\| < 1$, then,

$$\|\mathbf{\Sigma}^{1/2}\hat{\mathbf{\Sigma}}^{-1}\mathbf{\Sigma}^{1/2}\| \leq \frac{1}{1 - \|\mathbf{\Sigma}^{-1/2}(\hat{\mathbf{\Sigma}} - \mathbf{\Sigma})\mathbf{\Sigma}^{-1/2}\|}. \tag{47}$$

Lemma 3 (Extended regret error of empirical multi-output least square solution from [4] Theorem 11)**.** *Pick any $\delta \leq \min\{1, de^{-2.6}\}$, by Assumption 1 and Assumption 2, the mutually independent noise has same variance σ, with probability at least $1-\delta$, the following holds:*

$$\|\hat{\boldsymbol{A}}_{\text{inv}} - \boldsymbol{A}_{\text{inv}}\|_{\boldsymbol{\Sigma}}^2 \leq \frac{2d\sigma^2(d + 2\sqrt{d\ln(3/\delta)} + 2\ln(3/\delta))}{n} + o(1/n), \tag{48}$$

where $o(1/n)$ denotes a higher order term.

Proposition 1 (Error decomposition)**.** *Define the bias from the OOD, and the variance. Here let $\bar{\boldsymbol{A}}$ denote the conditional expectation of $\hat{\boldsymbol{A}}$ given $\boldsymbol{V}$:*

$$\varepsilon_{bs} := \|\bar{\boldsymbol{A}} - \boldsymbol{A}_{\text{inv}}\|_{\boldsymbol{\Sigma}}^2, \quad \varepsilon_{vr} := \|\hat{\boldsymbol{A}} - \bar{\boldsymbol{A}}\|_{\boldsymbol{\Sigma}}^2.$$

Follow the triangle inequality $(a+b)^2 \leq 2(a^2+b^2)$, the general error decomposition:

$$\|\hat{\boldsymbol{A}} - \boldsymbol{A}_{\text{inv}}\|_{\boldsymbol{\Sigma}}^2 \leq \varepsilon_{bs} + \varepsilon_{vr} + 2\sqrt{\varepsilon_{bs}\varepsilon_{vr}} \tag{49}$$

$$\leq 2(\varepsilon_{bs} + \varepsilon_{vr}). \tag{50}$$

Proof. The proof of Theorem 2 uses the decomposition of $\|\hat{\boldsymbol{A}} - \boldsymbol{A}_{\text{inv}}\|_{\boldsymbol{\Sigma}}^2$ in Proposition 1, and then bounds each term using the lemmas declared this section. The Eq. (6) can be translated into:

$$\mathcal{L}(\boldsymbol{A}) = \min_{\boldsymbol{A}} \sum_e \mathcal{L}_{\text{inv}}^e(\boldsymbol{A}) + h(\boldsymbol{A}) \tag{51}$$

$$= \min_{\boldsymbol{A}} \sum_e \hat{\mathbb{E}}^e \left[\|\boldsymbol{v} - \boldsymbol{A}\boldsymbol{v}\|^2\right] + h(\boldsymbol{A}) \tag{52}$$

Because $h(\boldsymbol{A})$ is a strict constraint, so the minimizer of each term would have $h(\boldsymbol{A}) = 0$, then we have:

$$\|\hat{\boldsymbol{A}} - \boldsymbol{A}_{\text{inv}}\|_{\boldsymbol{\Sigma}}^2 = \underbrace{\sum_e \hat{\mathbb{E}}^e \left[\|\boldsymbol{v} - \bar{\boldsymbol{A}}\boldsymbol{v}\|^2\right] - \mathbb{E}^e \left[\|\boldsymbol{v} - \boldsymbol{A}_{\text{inv}}\boldsymbol{v}\|^2\right]}_{\varepsilon_{bs}}$$
$$+ \underbrace{\sum_e \hat{\mathbb{E}}^e \left[\|\boldsymbol{v} - \hat{\boldsymbol{A}}\boldsymbol{v}\|^2\right] - \mathbb{E}^e \left[\|\boldsymbol{v} - \bar{\boldsymbol{A}}\boldsymbol{v}\|^2\right]}_{\varepsilon_{vr}}, \tag{53}$$

pick any $\delta \leq \min\{1, de^{-2.6}\}$, if the Assumption 1, Assumption 2 and Assumption 3 hold, with the probability at least $1-\delta$, if we have:

$$n > 6\rho^2 d(\ln(d) + \ln(3/\delta)), \tag{54}$$

the following holds:

1. Relatvie spectral norm error in $\hat{\boldsymbol{\Sigma}}$:

$$\|\boldsymbol{\Sigma}^{1/2}\hat{\boldsymbol{\Sigma}}^{-1}\boldsymbol{\Sigma}^{1/2}\| \leq \frac{1}{1-\kappa_s}, \tag{55}$$

   then, by Lemma 2:

$$\kappa_s := \sqrt{\frac{4\rho^2 d(\ln d + \ln(3/\delta))}{n}} + \frac{2\rho^2 d(\ln d + \ln(3/\delta))}{3n}. \tag{56}$$

2. Effect of bias:

$$\varepsilon_{bs} \leq \frac{2E}{(1-\kappa_s)^2}\left(\frac{\mathbb{E}\left[\|\boldsymbol{\Sigma}^{-1/2}\boldsymbol{v}(\boldsymbol{v} - \boldsymbol{A}\boldsymbol{v})\|^2\right]}{n}\left(1 + \sqrt{8\ln(3/\delta)}\right)^2 + \frac{16b_0^2 d\ln(3/\delta)^2}{9n^2}\right) \tag{57}$$

$$\leq \frac{2E}{(1-\kappa_s)^2}\left(\frac{\rho^2 d\mathbb{E}[\|\boldsymbol{v} - \boldsymbol{A}\boldsymbol{v}\|^2]}{n}\left(1 + \sqrt{8\ln(3/\delta)}\right)^2 + \frac{16b_0^2 d\ln(3/\delta)^2}{9n^2}\right). \tag{58}$$

3. Effect of noise:

$$\varepsilon_{vr} \leq \frac{1}{1-\kappa_s} \cdot \frac{2d\sigma^2(d+2\sqrt{d\ln(3/\delta)}+2\ln(3/\delta))}{E*|\mathcal{D}_L^e|}. \tag{59}$$

where the total parameters are trained in the mixture of domains, the training set can be regarded as one domain with $E*|\mathcal{D}_L^e|$.

As the $\mathbb{E}[v_i|\mathbf{Pa}_i] = \boldsymbol{A}_i\boldsymbol{v}$ is correct almost surely, and the term with b_0 appears only in the $o(1/n)$ term, simplify the representation:

$$\|\hat{\boldsymbol{A}} - \boldsymbol{A}_{\text{inv}}\|_{\boldsymbol{\Sigma}}^2 \leq \frac{2d\sigma^2(d+2\sqrt{d\ln(3/\delta)}+2\ln(3/\delta))}{E*|\mathcal{D}_L^e|} + o(1/n). \tag{60}$$

Comparing $\hat{\mathcal{L}}(\boldsymbol{A}_{\text{inv}})$ with $\hat{\mathcal{L}}(\boldsymbol{A})$. Assume the absolute value of each element is lower bounded as $|v_{y,i}| \geq \bar{v}_i$, the variance is lower bounded as $\text{Var}(v_{y,i}) \geq s$. For a set of domain-private (random) factors D_r, if $\boldsymbol{A}$ maintains them, by simple algebra, we have:

$$\sum_e \mathbb{E}^e\|[\boldsymbol{v}-\boldsymbol{A}\boldsymbol{v}]_{\mathbf{Pa}_y}\|^2 - \mathbb{E}^e\|[\boldsymbol{v}-\boldsymbol{A}_{\text{inv}}\boldsymbol{v}]_{\mathbf{Pa}_y}\|^2 = \sum_{i\in D_r} v_i \text{Var}(v_{y,i}), \tag{61}$$

Hence, with a probability of at least $1-\delta$, we have

$$\begin{aligned}
&\hat{\mathcal{L}}(\boldsymbol{A}) - \hat{\mathcal{L}}(\boldsymbol{A}_{\text{inv}}) && (62)\\
=&\text{Var}(\epsilon) + \sum_{i\in D_r} v_i \text{Var}(v_{y,i}) + \varepsilon_{bs}(\boldsymbol{A}) + \varepsilon_{vr}(\boldsymbol{A}) && (63)\\
&- \text{Var}(\epsilon) - \varepsilon_{bs}(\boldsymbol{A}_{\text{inv}}) - \varepsilon_{vr}(\boldsymbol{A}_{\text{inv}}) && (64)\\
\geq&\frac{1}{1-\kappa_s}\frac{\sigma^2(d+2\sqrt{d\ln(3/\delta)}+2\ln(3/\delta))}{E*|\mathcal{D}_L^e|} + \bar{v}s. && (65)
\end{aligned}$$

Further, one of the elements v_y in vector $\boldsymbol{v}$ is specified as the label, $\boldsymbol{W}$ is the ideal stable classifier parameter, and $\hat{\boldsymbol{W}}$ is an estimated classifier parameter. Then by the [4] Remark 12, if:

$$n \geq 6\rho^2|\mathbf{Pa}_y|(\log(|\mathbf{Pa}_y| + \ln(3/\delta)). \tag{66}$$

with the probability at least $1-\delta$, the following bound on the stable classifier parameters holds:

$$\|\hat{\boldsymbol{W}} - \boldsymbol{W}\|_{\boldsymbol{\Sigma}}^2 \leq \frac{2|\mathbf{Pa}_y|\sigma_y^2(|\mathbf{Pa}_y| + 2\sqrt{|\mathbf{Pa}_y|\ln(3/\delta)}+2\ln(3/\delta))}{E*|\mathcal{D}_L^e|} + o(1/n). \tag{67}$$

To simplify the condition formulation, we have the following:

$$n = E*|\mathcal{D}_L^e| > Q_1 + Q_2\ln(d/\delta), \tag{68}$$

where $Q_1 = 6\rho_y^2|\mathbf{Pa}_y|(\log(|\mathbf{Pa}_y| + \ln(3/\delta)) + 6\rho^2 d\ln(3)$, $Q_2 = 6\rho^2 d$. □

D. Additional Experimental Results

D.1. Complete Results on CMNIST

For all experiments on the CMNIST dataset, we report the accuracy of the last step. All reported statistics are average values over 3 random seeds. We set the max scale ratio of penalty weight as 10^6 for all the experiments and datasets.

In Figure D.1, we plot the accuracies of training on different DG subtasks on CMNIST along with the penalty during training. First, we can observe the accuracies of iDAG consistently approximate the oracle level in test domains across all subtasks. Second, the overall trends on three subtasks follow our arguments that the acyclicity penalty term increases as the model searches the invariant causal structure. And then with the convergence of the model reached, the penalty disappears and the performance becomes steady.

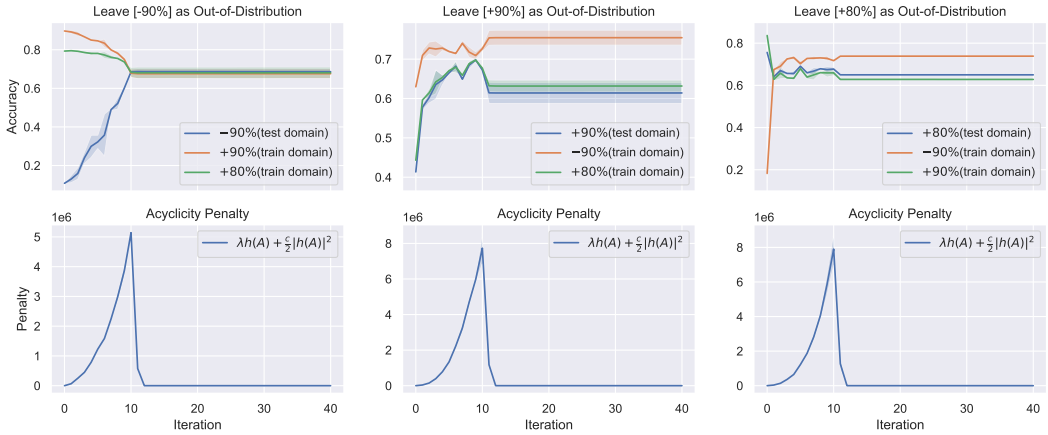

Figure D.1. Complete results on CMNIST

D.2. The Influence of Prototype-based Strategy

There are several ways to learn DAG and hence, we further test a variant of iDAG that directly optimizes the DAG module till it reaches the convergency within tolerance $\Delta\mathcal{L}_{\mathcal{G}} \leq 10^{-3}$ on the whole dataset features at the end of each epoch (dubbed iDAG (Feature-based)). We train the models using one Tesla V100 GPU respectively and evaluate the total training time. According to Table D.1, the

Table D.1. Training time and accuracy of different DAG optimization methods.

Method	Ar	Cl	Pr	Rw	Avg	Time (hour)
iDAG	68.2	57.9	79.7	81.4	71.8	8.46
iDAG (Feature-based)	63.2	55.9	77.7	79.4	69.0	24.32

variant that achieves competitive results is much slower than iDAG and difficult to tune due to the indeterminacy of DAG convergence.

D.3. Runtime Complexity Analysis

Table D.2. Time costs (hours)

Time	ERM	iDAG	$\mathcal{L}_{DAG}$
	8.26	8.46	0.07

In effect, iDAG is highly **time-efficient**. This is because the DAG constraint loss $\mathcal{L}_{DAG}$ is applied to updated prototypes post each mini-batch update, avoiding iterating the entire dataset. To substantiate this, we provide theoretical and empirical evidence: (1) **Theoretically**: iDAG's time complexity is $\mathcal{O}(NM + Cd^2)$, while ERM's is $\mathcal{O}(NM)$. Here, N, M denote the number of samples and the cost of updating the main network, and $C \ll N, d^2 \ll M$ represent the number of prototypes and parameters of the DAG matrix. (2) **Empirically**: As shown in Table D.2, the total runtime on the OfficeHome dataset reveals negligible time increases.

E. Implementation

In this section, we first provide the detail of augmented Lagrangian optimization of the acyclic constraint. Then, the network architecture details of iDAG. The hyperparameter selection criteria and ranges are presented. The training settings are summarized. Our code is mainly built on the SWAD [8] codebase with a hyperparameter searching strategy from DomainBed [18].

E.1. Acyclic Constraint

For achieving the acyclic constraint, following the concept that acyclic property is ensured by nodes can not reach itself up to infinite steps on the DAG $\boldsymbol{A} \in \mathbb{R}^{d+1\times d+1}$. There are multiple formulations of this constraint as,

$$h(\boldsymbol{A}) = \text{Tr}(e^{\boldsymbol{A}\odot\boldsymbol{A}}) - (d+1) = 0, \tag{69}$$

To incorporate this constraint into overall optimization, we rewrite the Eq. (6) and define the augmented Lagrangian,

$$\mathcal{L}_{\mathcal{G}} = \mathcal{L}_{\text{rec}} + \lambda h(\boldsymbol{A}) + \frac{c}{2}|h(\boldsymbol{A})|^2 + \lambda_{l1}|\text{vec}(\boldsymbol{A})|. \tag{70}$$

where λ is the Lagrangian multiplier and c is the penalty weight. When $c \to +\infty$, minimizing the $\mathcal{L}_{\mathcal{G}}$ is equivalent to the optimization on acyclicity solely. For stable training, the strategy is to progressively increase the c. A typical effective rule for updating the λ and increasing the c is,

$$(\boldsymbol{A}^{(t)}, \boldsymbol{\theta}^{(t)}) = \arg\min_{\boldsymbol{A},\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{A}, \boldsymbol{\theta}, \lambda^{(t)}, c^{(t)}), \tag{71}$$

$$\lambda^{(t+1)} = \lambda^{(t)} + c^{(t)} h(\boldsymbol{A}^{(t)}), \tag{72}$$

$$c^{(t+1)} = \begin{cases} \eta c^{(t)}, & \text{if} |h(\boldsymbol{A}^{(t)})| > \alpha |h(\boldsymbol{A}^{(k-1)})|, \\ c^{(t)}, & \text{otherwise}, \end{cases} \tag{73}$$

where t indicates the train epoch, $\alpha > 1$ indicates the scale ratio constant for increasing the c depends on whether optimized $h(\boldsymbol{A}^{(t)})$ less than a ratio of α. In general, we set $\eta = 10$ and $\alpha = 1/4$.

To incorporate the DAG learning with overall networks, we create a DAG layer following [10] by using two separate non-negative weight matrix $\boldsymbol{A}^-$ and $\boldsymbol{A}^+$. Then rewrite the DAG learning objective to

$$\min_{\boldsymbol{A}} \mathcal{L}_{\text{rec}}(\boldsymbol{A}) + \lambda_{l1}|\text{vec}(\boldsymbol{A})| \Leftrightarrow \min_{\boldsymbol{A}^-\geq 0, \boldsymbol{A}^+\geq 0} \mathcal{L}_{\text{rec}}(\boldsymbol{A}^+ - \boldsymbol{A}^-) + \lambda_{l1}\mathbf{1}^\top(\boldsymbol{A}^- + \boldsymbol{A}^+), \tag{74}$$

where $\mathbf{1}$ is a vector of all ones.

E.2. Network Architecture for CMNIST

Closely following [3, 30], we build MLPs as the backbone network for CMNIST experiments. The details of the model are summarized in Table E.1. Note the task on CMNIST is a binary classification problem, therefore, we use the binary cross-entropy loss for the reconstructed element which corresponds to the label.

Table E.1. Architecture details. BS: batch size, ReLU: Rectified Linear Unit.

Configuration	Description	Output
Input: $\boldsymbol{x}$	Observation images	BS $\times$ 14 $\times$ 14 $\times$ 2
Flatten	Flat the inputs	BS $\times$ 392
Linear	390 neurons, ReLU	BS $\times$ 390
Linear	390 neurons, ReLU	BS $\times$ 390
DAGLinear	390 neurons	BS $\times$ 390
Classifier	classifier	BS $\times$ 1

E.3. Network Architecture for Conventional Datasets

For conventional benchmarks, we utilize the pre-trained ResNet-50 as the backbone. Two classifiers are used during training, one for classifying the reconstruction on the label element, and the other for classifying the invariant features.

Table E.2. Hyperparameter search space comparison. U and list indicate Uniform distribution and random choice, respectively.

Condition	Parameter	Default value	Random distribution
ResNet	batch size	32	32
	learning rate	5e-5	$10^{\mathrm{U}(-5,-3.5)}$
	dropout	0	$[0.0, 0.1, 0.5]$
	weight decay	0	$10^{\mathrm{U}(-6,-2)}$
	λ_{max}	10^6	$10^{\mathrm{U}(5,8)}$
	DAG sparsity loss weight λ_{l1}	10^{-3}	$U(0.0, 0.1)$
	reconstruction loss weight λ_2	1.0	$U(0.01, 2.0)$
	prototype update ratio γ	0.99	$U(0.99, 0.999)$
	$\mathcal{L}_{\mathrm{CL}-\nu}$ trade-off	1.0	$U(0.1, 2.0)$
	$\mathcal{L}_{\mathrm{CL}-\mu}$ trade-off	1.0	$U(0.1, 2.0)$
MLP	batch size	64	64
	learning rate	1e-3	$10^{\mathrm{U}(-4,-2)}$
	dropout	0	$[0.0, 0.1, 0.5]$
	weight decay	0	$10^{\mathrm{U}(-6,-2)}$
	λ_{max}	10^8	$10^{\mathrm{U}}(5, 9)$

E.4. Data Augmentation Configuration.

In domain generalization, data augmentation plays an important role as a type of regularization method. Despite the fact that there are many data augmentation methods, the results of the DG task are promising. In order to ensure a fair comparison, we only utilize the data augmentations that are contained in SWAD [8]. Our data augmentation process is based on SWAD's technical specifications. We randomly cropped the images to retain between 70% and 100%. The horizontal flipping and color jittering were applied randomly with a magnitude of 0.3, and we also randomly apply a Grayscale on the original image with 10% probabilities.

E.5. Optimization Details

We use the Adam optimizer for all experiments except the synthetic dataset which is optimized by L-BFGS-B. For conventional big image datasets, all input images are resized to 224×224. In addition to the acyclic penalty applied on the adjacent matrix $\boldsymbol{A}$ we also employ additional parameter clamp operators after every optimizer step. For both $\boldsymbol{A}^-$ and $\boldsymbol{A}^+$, we clip them into range $[0, +\infty)$ and zero their diagonal. The reason is to ensure the acyclicity and force of the adjacency matrix are learned by reconstructing the element through elements other than itself.

For synthetic experiments, it is no need to learn an extra feature extraction network as factors are already given by definition. Therefore, we utilize a more efficient optimizer L-BFGS-B by directly indicating the parameter bounds. In practice, we utilize the L-BFGS-B optimizer implemented by *Scipy* library.

E.6. Hyperparameter Search Protocol

We describe the hyperparameters selection range in Table E.2. The training is following the fully automated sweep on all DG subtasks and model selection. We also searched the SWAD configuration according to the original paper settings [8] in order to better cooperate with the SWAD technology.

E.7. Infrastructures

For CMNIST datasets, every experiment is conducted on a single NVIDIA RTX2080Ti, Python 3.9.13, PyTorch 1.10.0, Torchvision 0.11.0, and CUDA 11.2.

For big image datasets, every experiment is conducted on a single NVIDIA Tesla V100, Python 3.10.4, PyTorch 1.11.0, Torchvision 0.12.0, and CUDA 10.2.

F. Pseudo-Code of iDAG

We summarize the pseudo-code of our iDAG method in Algorithm 1.

Algorithm 1: Pseudo-code of iDAG (one epoch).

Input: Training dataset $\mathcal{D}_{\text{tr}} = \{\mathcal{D}_L^e\}_{e=1}^{n_e}$, featurizer ϕ, classifier ω, adjacent matrix $\boldsymbol{A} \in \mathbb{R}^{d+1\times d+1}$, in-domain class prototypes $\mathcal{B}_\nu = \{\boldsymbol{\nu}_i^e\}$ $(1 \le i \le C, 1 \le e \le n_e)$,cross-domain class prototypes $\mathcal{B}_\mu = \{\boldsymbol{\mu}_j\}$ $(1 \le j \le C)$.
for $iter = 1, 2, \ldots,$ **do**
  sample a mini-batch B from $\mathcal{D}_{\text{tr}}$
  // obtain pairwise total effect matrix
  $\boldsymbol{P}^{\text{tol}} = \left[\sum_{k=0}^{\infty} \frac{1}{k!}(\boldsymbol{A} \odot \boldsymbol{A})^k\right] = e^{\boldsymbol{A}\odot\boldsymbol{A}}$
  // collect raw bottleneck features
  $\mathcal{B}_z = \{\boldsymbol{z}^e = \phi(\boldsymbol{x}^e) | \boldsymbol{x} \in B\}$
  // collect invariant bottleneck features
  $\mathcal{B}_y = \{\boldsymbol{z}_y^e = [\boldsymbol{P}_{d+1,:d}^{\text{tol}}]^\top \odot \boldsymbol{z}^e | \boldsymbol{z}^e \in \mathcal{B}_z\}$
  for $\boldsymbol{z}^e \in \mathcal{B}_z, \boldsymbol{z}_y^e \in \mathcal{B}_y$ **do**
    // momentum in-domain prototype updating
    $\boldsymbol{\nu}_c^e = \text{Normalize}(\gamma\boldsymbol{\nu}_c^e + (1-\gamma)\boldsymbol{z}^e)$
    // momentum cross-domain prototype updating
    $\boldsymbol{\mu}_c = \text{Normalize}(\gamma\boldsymbol{\mu}_c + (1-\gamma)\boldsymbol{z}_y^e)$
    // in-domain positive set generation
    $\mathcal{P}_z(\boldsymbol{z}^e) = \{\boldsymbol{k}' | \boldsymbol{k}' \in \mathcal{B}_\nu, e' = e \wedge c' = c\}$
    // cross-domain positive set generation
    $\mathcal{P}_y(\boldsymbol{z}_y^e) = \{\boldsymbol{k}' | \boldsymbol{k}' \in \mathcal{B}_\mu, c' = c\}$
  end
  // construct factors
  $\mathcal{V} = \{(\boldsymbol{v}_c^e = \text{Concat}(\boldsymbol{\nu}_c^e, y), y) | \boldsymbol{\nu}_c^e \in \mathcal{B}_\nu, y = c\}$
  // DAG reconstruction loss
  $\mathcal{L}_{\text{rec}} = \sum_{(\boldsymbol{v},y)\in\mathcal{V}} \|[\boldsymbol{A}\boldsymbol{v}]_{:d} - \boldsymbol{v}_{:d}\| + \ell\left(\boldsymbol{w}^\top(\boldsymbol{A}_{d+1}^\top \odot \boldsymbol{v}), y\right)$
  // complete DAG loss
  $h(\boldsymbol{A}) = \text{Tr}(e^{\boldsymbol{A}\odot\boldsymbol{A}}) - (d+1)$
  $\mathcal{L}_{\mathcal{G}} = \mathcal{L}_{\text{rec}} + \lambda h(\boldsymbol{A}) + \frac{c}{2}|h(\boldsymbol{A})|^2 + \lambda_{l1}|\text{vec}(\boldsymbol{A})|$
  // in-domain contrastive loss calculation
  $\mathcal{L}_{\text{CL-}\nu}(\phi; \tau, \mathcal{P}_z, \mathcal{B}^e = \mathcal{B}_\nu^e \cup \mathcal{B}_z) = \frac{1}{|\mathcal{B}_z|}\sum_{\boldsymbol{z}\in\mathcal{B}_z}\left\{-\frac{1}{|\mathcal{P}_z(\boldsymbol{z})|}\sum_{\boldsymbol{k}_+\in\mathcal{P}(\boldsymbol{z})}\log\frac{\exp(\boldsymbol{z}^\top\boldsymbol{k}_+/\tau)}{\sum_{\boldsymbol{k}'\in\mathcal{B}^e(\boldsymbol{z})}\exp(\boldsymbol{z}^\top\boldsymbol{k}'/\tau)}\right\}$
  // cross-domain contrastive loss calculation
  $\mathcal{L}_{\text{CL-}\mu}(\phi; \tau, \mathcal{P}_y, \bar{\mathcal{B}} = \mathcal{B}_\mu \cup \mathcal{B}_y) = \frac{1}{|\mathcal{B}_y|}\sum_{\boldsymbol{z}\in\mathcal{B}_y}\left\{-\frac{1}{|\mathcal{P}_y(\boldsymbol{z})|}\sum_{\boldsymbol{k}_+\in\mathcal{P}(\boldsymbol{z})}\log\frac{\exp(\boldsymbol{z}^\top\boldsymbol{k}_+/\tau)}{\sum_{\boldsymbol{k}'\in\bar{\mathcal{B}}(\boldsymbol{z})}\exp(\boldsymbol{z}^\top\boldsymbol{k}'/\tau)}\right\}$
  // classification loss calculation
  $\mathcal{L}_{\text{cls}} = \mathbb{E}_{(\boldsymbol{x},y)\in\mathcal{D}_{\text{tr}}}\left[\ell(\omega(\phi(\boldsymbol{x}) \odot [\boldsymbol{P}^{\text{tol}}]_{d+1,:d}^\top), y)\right]$
  // network updating
  minimize loss $\mathcal{L} = \mathcal{L}_{\text{cls}} + \mathcal{L}_{\mathcal{G}} + \mathcal{L}_{\text{CL-}\nu} + \mathcal{L}_{\text{CL-}\mu}$
end
// update the Lagrangian parameter
$\lambda^{(t+1)} = \lambda^{(t)} + c^{(t)}h(\boldsymbol{A}^{(t+1)})$
if $|h(\boldsymbol{A}^{(t)})| > \alpha|h(\boldsymbol{A}^{(t-1)})|$ **then**
  $c^{(t+1)} = \eta c^{(t)}$
else
  $c^{(t+1)} = c^{(t)}$
end

Appendix References

[1] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant Risk Minimization. *ArXiv preprint*, 2019.

[2] Junbum Cha, Sanghyuk Chun, Kyungjae Lee, Han-Cheol Cho, Seunghyun Park, Yunsung Lee, and Sungrae Park. SWAD: Domain Generalization by Seeking Flat Minima. In *Proc. of NeurIPS*, 2021.

[3] Ishaan Gulrajani and David Lopez-Paz. In search of lost domain generalization. In *Proc. of ICLR*, 2021.

[4] Daniel Hsu, Sham M. Kakade, and Tong Zhang. Random Design Analysis of Ridge Regression. In *Proceedings of the 25th Annual Conference on Learning Theory*, 2012.

[5] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M. Hospedales. Deeper, broader and artier domain generalization. In *Proc. of ICCV*, 2017.

[6] Yong Lin, Hanze Dong, Hao Wang, and Tong Zhang. Bayesian Invariant Risk Minimization. In *Proc. of CVPR*, 2022.

[7] Ignavier Ng, AmirEmad Ghassami, and Kun Zhang. On the role of sparsity and DAG constraints for learning linear dags. In *Proc. of NeurIPS*, 2020.

[8] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *Proc. of ICCV*, 2019.

[9] Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep hashing network for unsupervised domain adaptation. In *Proc. of CVPR*, 2017.

[10] Xun Zheng, Chen Dan, Bryon Aragam, Pradeep Ravikumar, and Eric P. Xing. Learning sparse nonparametric dags. In *Proc. of AISTATS*, 2020.